اصول پردازش تصویر

نیم‌سال اول ۱۳۹۹-۱۴۰۰

مدرس: دکتر مصطفی کمالی تبریزی

دانشکده علوم ریاضی

تمرین سری پنجم ـ سوال اول

نام و نام‌خانوادگی: سیدعلیرضا خادم

شماره دانشجویی: ۹۷۱۰۰۳۹۸

زمان حدودی اجرا: ۴ دقیقه و ۳۰ ثانیه

ابعاد تصویر source: $1200_{px} \times 630_{px}$

ابعاد تصویر target: $3973_{px} \times 2032_{px}$

موارد لازم.

برای اجرا لازم است تا تصاویر 1.source.jpg و 1.target.jpg در مسیر EX5_Q1/ قرار داشته باشد. همچنین در پیاده‌سازی این سوال از کتابخانه‌های cv2 ، numpy و scipy استفاده شده است که قبل از اجرا بایستی این کتابخانه‌ها روی سیستم شما نصب باشد.

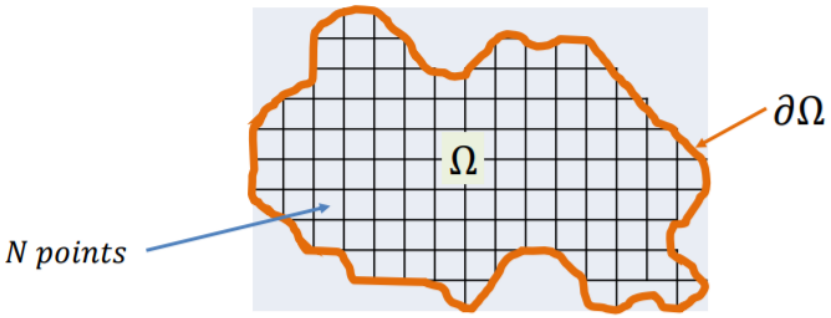
روند کلی حل.

ایده کلی حل به این صورت است که مسئله را به حلِ معادله پوآسون دوبعدی با شرایط مرزی دیریکله تبدیل می‌کنیم. با توجه به این نکته که گرادیان در تصاویر اهمیت زیادی دارد و در واقع آنچه میتوان از تصاویر درک کرد تا حد زیادی به گرادیان تصویر وابسته است، هدف ما این است که گرادیانِ تصویر souceای که میخواهیم در تصویر target قرار دهیم حفظ شود. استفاده‌ای که می‌خواهیم از گرادیان در حل این مسئله بکنیم به این صورت است که blending را به گونه‌ای انجام بدهیم گرادیان تصویر source بعد از قرار گرفتن در تصویر target همچنان حفظ شود. برای انجام این کار لازم نیست که کاربر object مدنظر از تصویر source را به صورت دقیق segment کند بلکه بهتر از که مقداری از background را هم داشته باشیم. (برای مثال کافیست که یک کانتور دورِ object کشیده شود.) کاری که میخواهیم انجام بدهیم به این صورت است که این کانتور را از تصویر source برداریم و در تصویر target قرار دهیم به طوری که گرادیان در ناحیه‌یِ داخل contour حفظ شود. خلاصه‌ای از روابطی که بایستی در این مسئله حل کنیم به صورت زیر است.

$$\begin{cases} \nabla^2 f(x,y) = g(x,y) & (x,y) \in \Omega \\ f(x,y) = h(x,y) & (x,y) \in \partial\Omega \end{cases} \qquad \nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4\, f_{i,j} = g_{i,j}$$

$$N \text{ unknowns}, N \text{ equations}$$
$$A\, f = b$$
$$\text{system of linear equations}$$



همانطور که در روابط بالا نشان داده شده است از روشِ عددیِ Finite Difference Method برای حل معادله پوآسون استفاده میکنیم. برای حل دستگاه معادله‌ای که در طی روند حل به آن برخواهیم خورد هم با توجه به این نکته که ماتریس ضرایب تعداد زیادی صفر دارد از کتابخانه scipy.sparse برای حل این دستگاه استفاده می‌کنیم.

توضیح کد.

برنامه در مجموع حاوی ۲ فایل با فرمت py. می‌باشد که توضیحات هر فایل در پایین آمده است.

○ utilities.py

warp_source_image(source_image, delta_x, delta_y, shape)

این تابع تصویر source و مقدار جابه‌جایی در راستای x و y و ابعاد تصویر target را به عنوان ورودی می‌گیرد و در خروجی تصویر با ابعاد shape برمی‌گرداند که در آن تصویر source با یک translation در مکان درست قرار گرفته است.

blend(warped_source, target, mask)

این تابع تصویر وارپ‌شده‌ی source ، تصویر target و mask را به عنوان ورودی می‌گیرد و بعد از اینکه ماتریس ضرایب A را با استفاده از تابع get_mat_A محاسبه کرد یک حلقه را اجرا می‌کند و برای هر channel ، blending را انجام می‌دهد.

get_mat_A(mask, shape):

این تابع برای محاسبه‌ی ماتریس ضرایب استفاده می‌شود. mask و shape را به عنوان ورودی می‌گیرد. در ابتدا با استفاده از تابع get_laplacian_matrix ، A را برابر با یک ماتریس پوآسون با ابعاد shape قرار می‌دهد و در ادامه درایه‌هایی از A که خارج از ناحیه سفید رنگِ mask قرار دارد را برابر با همانی قرار می‌دهد. در نهایت هم با استفاده از تابع tocsc آن را به صورت csc در می‌آوریم تا با استفاده از کتابخانه scipy.sparse.linalg بتوان دستگاه معادله را حل کرد.

$$A = \begin{bmatrix} D & -I & 0 & 0 & 0 & \dots & 0 \\ -I & D & -I & 0 & 0 & \dots & 0 \\ 0 & -I & D & -I & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -I & D & -I & 0 \\ 0 & \dots & \dots & 0 & -I & D & -I \\ 0 & \dots & \dots & \dots & 0 & -I & D \end{bmatrix},$$

I is the $m \times m$ identity matrix, and D, also $m \times m$, is given by:

$$D = \begin{bmatrix} 4 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 4 & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 4 & -1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -1 & 4 & -1 & 0 \\ 0 & \dots & \dots & 0 & -1 & 4 & -1 \\ 0 & \dots & \dots & \dots & 0 & -1 & 4 \end{bmatrix},$$

روابط دقیقتر را میتوان از این لینک مشاهده کنید.

get_mat_b(A, source_flat, target_flat, mask_flat)

این تابع ورودی‌هایی را که مشاهده می‌کنید را به عنوان ورودی می‌گیرد و b را ($Af = b$) محاسبه می‌کند و به عنوان خروجی برمی‌گرداند.

scale(image, min, max)

این تابع یک تصویر را به عنوان ورودی می‌گیرد و مقدار intensity پیکسل‌هایی که intensity آنها کمتر از مقدار min است را برابر و مقدار intensity پیکسل‌هایی که intensity آنهابیشتر از max است را برابر max قرار می‌دهد. در نهایت هم فرمت را به uint8 تغییر داده و به عنوان خروجی برمی‌گرداند.

blend_channel(A, source, target, mask, channel, shape)

این تابع ورودی‌هایی که مشاهده می‌کنید را به عنوان ورودی می‌گیرد و بعد از اینکه target[:, :, source[:, :, channel] و mask را به صورت flatten در آورد، با استفاده از تابع get_mat_b مقدار b را محاسبه می‌کندو در ادامه با استفاده از تابع spsolve از کتابخانه scipy.sparse.linalg دستگاه معادله $Af = b$ را حل می‌کند. در نهایت هم بعد از reshape کردن و scale کردنِ f ، آن را به عنوان خروجی بر می‌گرداند.

○ q۱.py

در این فایل ابتدا تصاویر را لود می‌کنیم و در ادامه result را با استفاده از تابع blend محاسبه می‌کنیم و با نام res1.jpg در مسیر EX5_Q1/results ذخیره می‌کنیم.